Graph Convolution Networks

Arka Sadhu

IIT Bombay

September 1, 2017

Outline

Outline

Introduction to Graph Convolutional Networks

- CNN are extremely efficient architectures for image and audio classification tasks.
- But CNN donot directly generalize to irregular domains such as graph.

Introduction to Graph Convolutional Networks

- CNN are extremely efficient architectures for image and audio classification tasks.
- But CNN donot directly generalize to irregular domains such as graph.
- Want to generalize CNN to Graphs.

Introduction to Graph Convolutional Networks

- CNN are extremely efficient architectures for image and audio classification tasks.
- But CNN donot directly generalize to irregular domains such as graph.
- Want to generalize CNN to Graphs.
- Non-trivial because the distances are non-euclidean.

Outline

Extending Convolutional to Graphs

There are two main approaches

- Spatial Approach :
  Generalization of CNN in the spatial domain itself.

Extending Convolutional to Graphs

There are two main approaches

- Spatial Approach :
  Generalization of CNN in the spatial domain itself.
  - Learning Convolutional Neural Networks for Graphs [ICML 2016].[1]

Extending Convolutional to Graphs

There are two main approaches

- Spatial Approach :
  Generalization of CNN in the spatial domain itself.
  - Learning Convolutional Neural Networks for Graphs [ICML 2016].[1]
- Spectral Approach :
  Using the frequency characterization of CNN and using that to generalize to Graphical domain

Extending Convolutional to Graphs

There are two main approaches

- Spatial Approach :
  Generalization of CNN in the spatial domain itself.
  - Learning Convolutional Neural Networks for Graphs [ICML 2016].[1]
- Spectral Approach :
  Using the frequency characterization of CNN and using that to generalize to Graphical domain
  - Spectral Networks and Deep Locally Connected Networks on Graphs [Bruna et al. ICLR 2014].
  - Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering [Defferrard et al. NIPS 2016] (will be the main focus)
  - Semi-Supervised Classification with Graph Convolutional Networks [Kipf et al. ICLR 2017]

Limitations of Spatial Approach

- Can't exactly define a neighborhood because the distances are not uniform.
- Ordering of nodes is problem specific.

Hence for the remainder we discuss the Spectral Approach

Outline

A Basic Formulation

- Convolution in spectral (Fourier) domain is point wise multiplication.
- Fourier Basis is defined as the eigen basis of the laplacian operator.
- Can use Laplacian of a graph.

Outline

Defining the Problem on Graphs

- A feature description x_i for every node i; summarized in a NxD feature matrix X (N : number of nodes, D : number of input features)

Defining the Problem on Graphs

- A feature description x_i for every node i; summarized in a NxD feature matrix X (N : number of nodes, D : number of input features)
- Adjacency Matrix A.

Defining the Problem on Graphs

- A feature description x_i for every node i; summarized in a NxD feature matrix X (N : number of nodes, D : number of input features)
- Adjacency Matrix A.
- Node level output Z (an NxF feature matrix, where $F =$ number of output features per node).

Outline

Brief overview of Graph Laplacian

Let T denote the diagonal matrix with (v,v)-th entry having value d_v : degree of vertex v. Define L-matrix as

$$L(u,v) = \begin{cases} d_v & \text{if u = v} \\ -1 & \text{if u and v are adjacent} \\ 0 & \text{otherwise} \end{cases}$$

And the Laplacian of the graph as

$$\mathcal{L}(u,v) = \begin{cases} 1 & \text{if u = v} \\ -\frac{1}{\sqrt{d_u d_v}} & \text{if u and v are adjacent} \\ 0 & \text{otherwise} \end{cases}$$

Graph Laplacian (contd.)

$$\mathcal{L} = T^{-1/2} L T^{1/2}$$

With the convention $T^{-1}(v, v) = 0$ for $d_v = 0$.
When G is k-regular,

$$\mathcal{L} = I - \frac{1}{k} A$$

For a general graph

$$\mathcal{L} = I - T^{-1/2} A T^{1/2}$$

Outline

Graph Fourier Transform

- Laplcian of the graph is real symmetric positive semidefinite, and thus can be written as
$$L = U \Lambda U^T$$
- Here $U = [u_0 .... u_{n-1}]$ is the fourier basis and $\Lambda = diag([\lambda_0 ... \lambda_{n-1}])$ are ordered real non-negative eigen values.
- Graph Fourier Transform of a signal x is $\hat{x} = U^T x$.

Spectral filtering of graph signals

- Defining convolution on graphs

$$x *_G y = U((U^T x) \odot (U^T y))$$

- Filtering by g_θ

$$y = g_\theta(L)x = g_\theta(U\Lambda U^T)x = Ug_\theta(\Lambda)U^T x$$

- A non-parametric filter (all parameters free) would be defined as

$$g_\theta(\Lambda) = diag(\theta)$$

Polynomial Parametrization

- Problem with non-parametric filters is that not localized (we want something like k-neighborhood) and therefore their learning complexity becomes $O(n)$. This can be overcomed with use of a Polynomial filter
$$g_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k \Lambda^k$$
- The advantage we gain here is that nodes which are at a distance greater than K away from the node i, at which the filter is applied, are not affected. Hence we have gained localization.

Recursive formulation for fast filtering

- Still cost to filter is high $O(n^2)$ because of multiplication with U matrix.
- Therefore use recurrence relation of chebyshev polynomial instead.

$$g_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k T_K(\tilde{\Lambda})$$

Here $\tilde{\Lambda}$ is scaled between $[-1, 1]$.

- This allows us to compute $\bar{x_k} = T_K \tilde{L} x$. And Therefore

$$y = g_\theta(L)x = [\bar{x_0} \ldots \bar{x_{k-1}}]\theta$$

- The cost is now $O(K|E|)$

Learning filters

- Trivial to show that backprop calculation can be done efficiently.

M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," *CoRR*, vol. abs/1605.05273, 2016.